Project 1 – A Software Day

Design

The Employee class formed a parent class for the workers at the company and implemented the primary daily flow of the workday. Developers, Team leaders, and Managers inherit from the Employee class and implement specific aspects of their workday.

The Manager class holds the conference room instance to ensure there is only one and provides access to its lock to the Team Leads. Threads that need to interact with the Manager, call methods provided by the PM class and their flow is directed by the PM.

Concurrency is controlled using the Reentrant lock and Conditions created from it.

Alternatives

The question asking requirement was not properly fulfilled, but were there more time to work on the project I would have considered using a separate thread to accomplish this in a more robust manner. It seemed that there was no 'simple' way to incorporate the necessary flow to allow a team member to ask questions at *any* time of the day that they were not in a meeting. It seemed like it would be better to have an external thread running for the team member that attempted to obtain a lock from the Employee and then if successful, incite the worker to ask a question.

Developers were also not implemented properly in terms of going to meetings. In the requirements it is stated that they should go to the conference room and wait for the room to be available. I was not able to find a way to properly communicate between the Team Lead and Developer to cause this to happen. The Team Lead has references to the Developers on their team, yet when I attempted to call methods in the developers, the Team Lead would lock and only the first Developer would get the message. I could only assume this meant that the Team Lead thread was also put into a wait state when it was called in the Developer. Perhaps I am not fully in the Thread mindset yet, but I could not find another way to accomplish this so I simply called a log message in the Developer to fake their cooperation.

The final meeting of the day was also not implemented but I figure this would be accomplished in the same way that the other meetings were implemented. I feel this could be done easily with a little more time available.

Statistics were not gathered on the time spent on each task for each Employee, also due to time constraints. Assuming all the other functionality was in place this would be a relatively easy task and will be implemented if additional time is available.

Testing

Not much testing was done other than the usual trial runs as functionality was introduced. I did notice that the flow of the workday was unique each time as expected since there is a random lateness to the arrival of each Employee. This is excluding the clock out times for lunch and end of day which were hard coded.

Final Thoughts

I am disappointed that I was not able to complete all the requirements for this project. It is rare that I find myself forced to hand in an assignment late *and* incomplete. I find concurrency a fun and challenging aspect of software development and would like to come away from each project with a full understanding of the concepts involved. Unfortunately this is not always possible to accomplish and sometimes projects fail, as we know. If time permits I will try to expand on this project and complete more, or hopefully all, of the requirements set forth.